PCT/IL 2004/000575
10 AUG 2004

PA 1195923

THE UNITED STATES OF AMERICA

TO ALL TO WHOM THESE PRESENTS SHALL COME:

UNITED STATES DEPARTMENT OF COMMERCE

United States Patent and Trademark Office

July 19, 2004

THIS IS TO CERTIFY THAT ANNEXED HERETO IS A TRUE COPY FROM THE RECORDS OF THE UNITED STATES PATENT AND TRADEMARK OFFICE OF THOSE PAPERS OF THE BELOW IDENTIFIED PATENT APPLICATION THAT MET THE REQUIREMENTS TO BE GRANTED A FILING DATE UNDER 35 USC 111.

APPLICATION NUMBER: *60/484,294*
FILING DATE: *July 03, 2003*

PRIORITY DOCUMENT
SUBMITTED OR TRANSMITTED IN COMPLIANCE WITH RULE 17.1(a) OR (b)

REC'D 27 AUG 2004
WIPO PCT

BEST AVAILABLE COPY

By Authority of the
COMMISSIONER OF PATENTS AND TRADEMARKS

T. Lawrence

T. LAWRENCE
Certifying Officer

PATENT APPLICATION SERIAL NO. ______________________

U.S. DEPARTMENT OF COMMERCE
PATENT AND TRADEMARK OFFICE
FEE RECORD SHEET

07/08/2003 RMEBRAHT 00000034 062140 60484294
01 FC:2005 80.00 DA

PTO-1556
(5/87)

*U.S. Government Printing Office: 2001 — 481-697/59173

07/03/03 198 U.S. PTO

Please type a plus sign (+) inside this box → +

Docket Number:	257514

PROVISIONAL APPLICATION FOR PATENT COVER SHEET

This is a request for filing a PROVISIONAL APPLICATION FOR PATENT under 37 CFR 1.53 (c).

17507 U.S. PTO 60/484294 07/03/03

INVENTOR(S)/APPLICANT(S)

Given Name (first and middle [if any])	Family Name or Surname	Residence (City and either State or Foreign Country)
ZOHAR	PELEG	RAANANA, ISRAEL

☐ Additional inventors are being named on page 2 attached hereto

TITLE OF THE INVENTION (280 characters max)

METHOD AND APPARATUS FOR PARTITIONING ALLOCATION AND MANAGEMENT OF JITTER BUFFER MEMORY FOR TDM CIRCUIT EMULATION APPLICATION

CORRESPONDENCE ADDRESS

Direct all correspondence to:

☐ Customer Number → Place Customer Number Bar Code Label here

OR

☐ Firm or Individual Name	Mark M. Friedman				
Address	c/o DISCOVERY DISCOVERY				
Address	9003 FLORIN WAY				
City	UPPER MARLBORO	State	MD	ZIP	20772
Country	US	Telephone	301-952-1011	Fax	301-952-9023

ENCLOSED APPLICATION PARTS *(check all that apply)*

☒ Specification	*Number of Pages*	15	Applicant is Small Entity
☒ Drawing(s)	*Number of Sheets*	5	☐ Other (specify)

METHOD OF PAYMENT OF FILING FEES FOR THIS PROVISIONAL APPLICATION FOR PATENT *(check one)*

☐ A check or money order is enclosed to cover the filing fees

☒ The Commissioner is hereby authorized to charge filing fees or credit any overpayment to Deposit Account Number: 06-2140

FILING FEE AMOUNT: $80

The invention was made by an agency of the United States Government or under a contract with an agency of the United States Government.

☒ No.

☐ Yes, the name of the U.S. Government agency and the Government contract number are:

Respectfully submitted,

SIGNATURE ______ DATE 30 JUN 03

TYPED or PRINTED NAME Mark M. Friedman

REGISTRATION NO. *(if appropriate)* 33,883

TELEPHONE (703) 415-1581

USE ONLY FOR FILING A PROVISIONAL APPLICATION FOR PATENT

SEND TO: Box Provisional Application, Assistant Commissioner for Patents, Washington, DC 20231

P19LARGE/REV04

APPLICATION FOR PATENT

Inventor: Zohar Peleg

Title: Method and apparatus for partitioning allocation and management of jitter buffer memory for TDM circuit emulation applications

FIELD AND BACKGROUND OF THE INVENTION

The present invention relates generally to the field of transmission of timing-sensitive synchronous data over asynchronous media, and in particular to the transmission of time division multiplexed (TDM) circuits over packet-switched networks (PSNs). More particularly, the present invention relates to jitter buffers, which are a key component in the reception of TDM payloads transmitted over a PSN. A jitter buffer is a device that supports a smooth play-out of synchronous, timing-sensitive data (e.g. audio, video, or TDM circuits) in cases where the data is received with jitter, due to propagation through asynchronous media, such as PSNs.

Emulation of a TDM circuit is done by sampling the TDM traffic, and de-multiplexing it into distinct channels. The data stream of each channel is sliced into fragments. Each fragment is encapsulated within a set of network headers, to form a packet, and transmitted over the packet network. Each TDM data fragment makes a payload of one packet. The TDM circuit is reconstructed at the receiving end, by extracting the packet payload, reassembling the channel data stream, and multiplexing the multiple channels data into a single TDM circuit. The delay between transmission and reception may vary from packet to packet. The variation in delay is referred to as "jitter". When the jitter is larger than the original time-interval between consecutive packets, packets may arrive out-of-sequence. Since the data must be played-out at the same rate and in the same order as the data has been sampled, the jitter must be removed, and packets must be reordered before play-out. The jitter removal and packet reorder are done by the jitter buffer.

SUMMARY OF THE INVENTION

The present invention is of a method and apparatus for partitioning allocation and management of jitter buffer memory, that supports multiple channels of TDM circuit emulation over packet networks, where each channel can run at different bit rate and different packet rate.

According to the present invention there is provided a method for partitioning allocation and management of jitter buffer memory for TDM circuit emulation applications comprising the steps of obtaining a channel hierarchy for a plurality of packet carrying channels having different channel rates, obtaining a packet sequential number, and using the channel hierarchy and the packet sequential number, generating a base address in the jitter buffer memory, whereby each channel is allocated a space in the buffer memory that is proportional to the rate of the channel, and whereby out-of-order packets are automatically reordered by the jitter buffer.

According to the present invention there is provided a method for partitioning a jitter buffer memory receiving a plurality of packet streams, each stream carried by a channel arranged in a given channel hierarchy, the method comprising the steps of: dividing the buffer memory into a plurality of hierarchically arranged queues, and allocating each queue to one channel, so that the queue hierarchy follows the channel hierarchy.

According to the present invention there is provided a hierarchically partitioned jitter buffer memory comprising a plurality of hierarchically arranged queues correlated with a channel hierarchy, and means for addressing said queues.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention is herein described, by way of example only, with reference to the accompanying drawings, wherein:

FIG. 1 shows the structure of a jitter buffer **100** according to the present invention;

FIG. 2 FIG. 2 shows an example of a possible OC-12 topology tree, which comprises a mixture of STS-3c, STS-1, VT-1.5 and VT-2 channels;

FIG. 3 shows an exemplary memory partitioning for the channel-tree hierarchy example shown in FIG. 2;

FIG. 4 is a flow chart describing schematically the address generation method according to the present invention;

FIG. 5 is a flow chart describing schematically an exemplary jitter buffer system implementation.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

The present invention is of a method and apparatus for partitioning allocation and management of jitter buffer memory for TDM circuit emulation applications. FIG. 1 shows the structure of a jitter buffer **100** according to the present invention. Jitter buffer **100** comprises a write logic block **102**, a memory **104**, a read logic **106**, a pre-fetch buffer **108**, and a buffer utilization monitoring unit **110**. Memory **104**, which is used for storing the received payload, is the core element of the jitter buffer. It may be any type of memory, for example SRAM, DRAM, SDRAM, RDRAM, etc., configurable in size from 256KB to 32MB, and operative to handle a compressed payload with a variable packet length (the variable length being the result of the compression). A main innovative feature of the jitter buffer memory according to the present invention is its capability to be optimized for TDM emulation by a hierarchical partitioning which follows the SONET/SDH hierarchy. The stored data is sorted into channels, where the data of each channel is ordered sequentially. That is, the sorted data is stored in the order it was transmitted, which is not necessarily the order it was received in. For example, suppose packets 1-2-3 were transmitted in this sequence but received in a sequence 1-3-2. When packet 3 is received it is not stored right after packet 1, but leaves empty space for packet 2. When packet 2 is received it is not stored after packet 3, but in the empty space saved for it between packets 1 and 3.

Write logic **102** receives a packet, detects the destination circuit, verifies that the packet is within the desired, programmable time window, and generates a write address in order to place the packet in the right order within the right channel queue. Then it stores the packet payload sequentially into this memory location (i.e. the range of addresses starting at the base address as above). While the TDM data needs to be played out one-byte at a time in accurate timing, the jitter buffer memory is accessed

with long bursts and wider data path, and requires large arbitration and access time, relative to the play-out timing granularity. Pre-fetch buffer **108** is designed to guarantee consistent data flow, while compensating for the differences between the memory access and data play-out. Buffer **108** includes a small temporary buffer per-channel with allocated size which is proportional to the channel's rate. The temporary buffer is filled by reading multiple words of data from the jitter-buffer memory, using burst-read access, and dispenses single bytes into the TDM stream, in the right time-slots. The pre-fetch buffer is designed to start fetching early enough, before the buffer empties or drains-out, and to fetch and hold enough data so that the TDM play-out is never starved.

Read logic **106** performs the read access as requested by the pre-fetch buffer, while tracking the read pointer of each channel. Buffer utilization monitoring unit **110** holds a buffer-utilization counter per channel, which provides the instantaneous number of bytes stored in the queue at any given moment. The buffer utilization monitoring unit monitors the amount of buffered data in each channel, by adding the amount of data that is written into the buffer and subtracting the amount of data played-out of the buffer.

Each channel can be in either one of two jitter-buffer states, "fill" or "normal". In the "fill" state, data is written to the buffer but there is no TDM data play-out. The buffer utilization is monotonously increasing. When the channel reaches its pre-determined operating point, it moves into a "normal" state, in which the data is read from the jitter-buffer queue and played-out on the TDM circuit. As long as the play-out rate is synchronized with the sampling rate at the transmitting end, the average utilization remains balanced around the operating point. The operating point is a programmable depth of the jitter buffer. If the transmission is disrupted for a period long enough to drain the buffer, the buffer empties and returns to the "fill" state.

TDM Channel Hierarchy

In order to better understand the method of the present invention, reference is first made to the normal way of optimizing the jitter buffer structure for storing channelized SONET circuit payloads. The OC-12 circuit hierarchy comprises the following levels: each STS-12 contains 4 STS-3 circuits. Each STS-3 contains 3 STS-1 circuits (total of up to 12 STS-1 circuits in OC-12). Each STS-1 circuit contains 7

VTG circuits (total of up to 84 VTG circuits in OC-12). Each VTG circuit contains 4 VT-1.5 circuits or 3 VT-2 circuits (total of up to 336 VT-1.5 circuits or 252 VT-2 circuits in OC-12).

The SONET/SDH traffic may carry any mixture of the circuit levels. Going through circuit emulation over PSN, this traffic may be distributed into distinct channels, where each channel is packetized and transmitted as a separate packet-flow over the PSN. Some flexibility is provided in mapping of the TDM circuit structure into the PSN channel structure, since some channelized payload structures allow the choice of either mapping the circuit into a single channel or breaking it down to multiple lower-rate channels. The following table shows various options for mapping TDM circuits into PSN channels.

Table 1: Circuit to PSN-channel mapping options – Structured Mode

SONET Circuit	SDH Circuit	Possible PSN channels
Non-channelized STS-1 SPE	VC-3	STS-1/VC-3
Channelized STS-1 SPE	VC-3	STS-1 / VC-3 Fractional STS-1 / VC-3 VT-1.5 / VC-11 VT-2 / VC-12
STS-3c SPE	Non-channelized VC-4	STS-3c / VC-4
Channelized STS-3	STM-1 (3*AU-3)	STS-1 / VC-3 VT-1.5 / VC-11 VT-2 / VC-12
(No SONET Equivalent)	Channelized VC-4 (1*AU-4)	Fractional VC-4 VC-3, fractional VC-3 VT-1.5 / VC-11 VT-2 / VC-12
STS-12c-SPE	VC-4-4c	STS-12c / VC-4-4c
Channelized STS-12	STM-4	STS-3c / VC-4 STS-1 / VC-3

		VT-1.5 / VC-11 VT-2 / VC-12

FIG. 2 shows an example of a possible OC-12 topology tree, which comprises a mixture of STS-3c, STS-1, VT-1.5 and VT-2 channels. According to one aspect of the present invention, the tree configuration or topology is a set of configurable stop flags, which is used to select any subset of this tree, by selecting the level of the leaf-node on each branch. The tree-creation is used as an input to the method of the present invention, and other tree topologies may be used for the same purpose. A stop-flag per node determines that the corresponding node becomes a leaf (as long as the branch is not stopped at a higher level). Any node that resides below a leaf node would be trimmed-off and excluded from the selected tree, and the corresponding payload would remain multiplexed within the channel marked by the remaining leaf-node. Each incoming packet is uniquely identified and associated with the corresponding channel, allowing multiple channels, carried over multiple packet flows, to be multiplexed into a single TDM stream.

Channel naming convention

Each node on the channel-tree has a unique name (designation) of the form {Type (J, K, L, M)}. J designates the STS-3/VC-4 number (1 to 4) within the STS-12/STM-4 level. K designates the STS-1/TUG-3 or VC-3 number (1 to 3) within the STS-3/VC-4 level. L designates the VT-Group / TUG-2 number (1 to 7) within the STS-1/ TUG-3 level. M designates the VT-2 / TU-12 number (1 to 3) or the VT-1.5/TU-11 number (1 to 4) within the VT-group / TUG-2 level. A zero value indicates that the corresponding level is below the stop-level and therefore outside of the selected tree. Using this convention, the following channels are defined in the example shown in FIG. 2:

1. The first STS-3 circuit **304** is divided as follows: STS-1 circuit #1 **(302)** is broken down to 28 VT-1.5 channels, named VT1.5(1,1,1,1) **306**, VT1.5(1,1,1,2), ..., VT1.5(1,1,7,4) **308**. STS-1 circuit #2 **310** and STS-1 circuit

#3 312 are designated as channels, named STS-1(1,2,0,0) and STS-1(1,3,0,0).

2. The second STS-3 circuit **314** is designated as an STS-3c channel, named STS3(2,0,0,0).

3. The third STS-3 circuit **316** is divided as follows: the first STS-1 circuit **318** and the third STS-1 circuit **324** are designated as channels named STS1(3,1,0,0) and STS1(3,3,00) respectively. The second STS-1 circuit is divided into 21 VT-2 channels named VT2(3,2,1,1) **320** , …, VT2(3,2,7,3) **322.**

4. The fourth STS-3 circuit **326** is designated as an STS-3c channel, named STS3(4,0,0,0).

Buffer Partitioning and Memory allocation

The jitter buffer memory is hierarchically divided into queues, where each queue is allocated to one channel. A queue is a memory space designated for buffering the packet stream of one channel. The queue hierarchy follows the channel hierarchy, with the exception that the partitioning into queues is done by using powers of 2 division factors, to maintain an easy addressing scheme. For example, while there are 3 STS-1's in one STS-3, the STS-3 memory area is divided into 4 STS-1 queues, of which 3 are used by the 3 STS-1 channels and 1 remains unused, see below. When the jitter buffer memory is entirely allocated to a channelized STS-12 circuit, the jitter buffer memory is partitioned as follows: the entire memory may be used for one STS-12 channel, or further divided into 4 sections, where each section is designated for one STS-3 channel. Each STS-3 memory section may be entirely used for an STS-3 channel or further divided into 4 equal STS-1 sections, out of which 3 are in use and one is reserved. Each STS-1 memory section can be used for an STS-1 channel, or further divided into 8 equal VTG sections, out of which 7 are in use and one is reserved. Each VTG section is further divided into 4 VT-1.5 sections, out of which 3 are in use and one is reserved.

As a result, the memory hierarchy (as opposed to the actual circuit hierarchy) is divided into 512 VT-1.5 sections (out of which 336 are in use) or up to 16 STS-1

sections out of which 12 are in use, or up to 4 STS-3 sections or a single STS-12 section.

FIG. 3 shows an exemplary memory partitioning for the channel-tree hierarchy example shown in FIG. 2. As mentioned, a queue is a memory space designated for buffering the packet stream of one channel. Each queue is further divided into segments, where the number of segments and the size of each segment in bytes are both integer powers of 2 (i.e. 2^n where n is an integer number). Each segment is designed to hold one and only one packet, and therefore the size of the segment is determined as the minimum integer power of 2 that can hold the maximum packet size for the channel. Once the segment size is determined, the number of segments is determined as well, given a predetermined queue size.

For example, a 8MB memory is allocated to a channelized STS-3 circuit, and the third STS-1 circuit is further divided into 28 VT-1.5 channels, each with a packet-payload size of 27 bytes. The queue allocation is therefore 2MB queue for each STS-1 channel (1/4 of 8MB), and 64KB for each VT-1.5 channel (1/32 of 2MB). Due to this partition, the maximum queue size is roughly proportional to the rate of the channel, and the buffering capability is distributed evenly between channels of different rates, as demonstrated in the following.

With the partitioning in the example above, the STS-1 channel segment size is 1024 bytes, which is the minimum integer power of 2 required for holding 783 bytes, and therefore there are 2048 segments. The VT-1.5 segment size is 32 bytes, and therefore the number of segments per queue is also 2048 segments. Given that each packet in this example is holding one frame, which is time-equivalent data of 125μs, the 2048 segments of either channel type can provide up to 256ms of buffering time.

Each channel may be independently configured for using any portion of the allocated maximum buffering time by selecting the buffer depth from one packet to full queue size. Each arriving packet is uniquely associated with a unique channel identification (channel ID or CH-ID) and, in an exemplary case, a 14-bit sequential number. A 1:1 address mapping scheme is used to provide each arriving packet with a pre-allocated segment, as follows: the queue base address is a 1:1 mapping of the channel identification number, and the segment base address is a 1:1 mapping of the packet sequential number. This allocation method guarantees efficient and fast addressing and provides automatic reordering, since each arriving packet is stored in the right segment, regardless of its arrival order.

Address generation method

FIG. 4 is a flow chart describing schematically the base-address generation method according to the present invention (the address is combined of base-address which is the innovative part, and byte-offset which is standard). The base-address formation process preferably comprises the following steps:

1. Determining in a step **402** the number of address bits as a log-2 of the total memory size. In the following examples shown in Tables 2, 3, and 4 below, a 25-bit address is selected for supporting a 32MB memory.

2. Determining in a step **404** the upper 2, 4 or 9 bits using the channel identification number (ID) for STS-3, STS-1 or VT channels respectively: 2 bits if the channel is STS-3, 4 bits if the channel is STS-1 or 9 bits if the channel is VT.

3. Determining in a step **406** the number of lower-most bits allocated for the byte offset as the log 2 of the selected packet size.

4. Determining in a step **408** the number of bits to be copied from the lower side of the packet sequential number as the remaining bits (in the middle section of the address word), after subtracting the byte-offset bits and the channel ID bits from the total address size.

5. Selecting in a step **410** the effective channel ID bits from the input channel ID, by masking the irrelevant bits and by shifting the remaining bits to the right location in the address given by the left-most 2, 4, or 9 bits as determined by step **404**, starting at the highest bit location as determined by step **402**.

6. Selecting in a step **412** the effective part of the sequential number, by masking out the irrelevant bits and by shifting the remaining bits to the right location in the address word, which is immediately next to the channel ID bits.

Table 2 shows a few examples of address generation corresponding to several channel types with different packet sizes, assuming a 32MB memory divided into various queue types as shown in FIG. 3.

<table>
<tr><th>Ch #</th><th>Channel Type (Payload Size)</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>1</td><td>STS-12c (783)</td><td colspan="15">Sequential Number</td><td colspan="10">Byte offset</td></tr>
<tr><td>2</td><td>STS-3c (783)</td><td colspan="2">STS3#</td><td colspan="13">Sequential Number</td><td colspan="10">Byte offset</td></tr>
<tr><td>3</td><td>STS-1 (783)</td><td colspan="2">STS3#</td><td colspan="2">STS1 #</td><td colspan="11">Sequential Number</td><td colspan="10">Byte offset</td></tr>
<tr><td>4</td><td>STS-1 (261)</td><td colspan="2">STS3#</td><td colspan="2">STS1 #</td><td colspan="12">Sequential Number</td><td colspan="9">Byte offset</td></tr>
<tr><td>5</td><td>VT-1.5 (27)</td><td colspan="2">STS3#</td><td colspan="2">STS1#</td><td colspan="3">VTG#</td><td colspan="2">VT#</td><td colspan="11">Sequential Number</td><td colspan="5">Byte offset</td></tr>
</table>

Table 2

In the following example, Table 3, the memory allocation for STS-3c channel #2 (314 in FIG. 3) identified by (2,0,0,0) is done as follows: the queue is 8MB from address 0x800000 to address 0xFFFFFF. Bits 24-23 have a fixed value of 01 for this queue, and the remaining 23 bits are divided between the segment address and the byte offset as follows: the packet size is 783 bytes (1/3 frame), and therefore the packet segment is 1024 bytes. The lower 10 bits are therefore the byte offset within the packet segment, and the remaining 13 bits (22-10) are used as a segment address, determined by the 13 lower bits of the packet sequential number. The resulting address is

24-23	22 - 10	9 - 0
01	Lower 13 bits of sequential number	Byte offset within packet-payload

Table 3

In the following example in Table 4, the VT-1.5 channel is identified as 3-2-5-4 (STS-3 #3, STS-1 #2, VTG #5, VT-1.5 #4). The queue is 64KB (65536 bytes) at base address 0x1330000, allowing up to 2048 segments of 32-bytes each.

24-23	22-21	20-18	17-16	15 - 5	4 - 0
10	01	100	11	Lower 11 bits of sequential number	Byte offset

Table 4

Method of operation

The method of operation is shown a flow chart in FIG. 5. Upon an initialization step **502**, the jitter buffer is set into a "fill" state, and the buffer utilization counter is set to 0. When a new packet is received in a step **504**, a circuit multiplexing label in the packet header is used to associate each received packet with the corresponding channel. Once the channel is identified in a step **506**, the base address of the queue is automatically determined in a step **508** as described above. The jitter-buffer write logic keeps track of the allowed write-window per channel, which is a sliding window of sequential numbers that start one packet above the presently read packet, and ends at the maximum queue size above the presently read packet. The sequential number of each received packet is evaluated against the allowed write-window **510**. Packets that do not fit in the window are dropped. The window size can be further limited below the physical size of the memory allocated for the queue, in order to increase the filtering of stale or bad packets. If the packet fits in the write-window, the segment address is determined as a function of the sequential number in a step **512**, as described above, and the packet is written into the corresponding queue in a step **514**. The buffer utilization counter is incremented in a step **516**. Testing the condition of "playout-enable" in a step **518** reveals if the buffer is in the fill state or in the normal state. While in the fill state, queued packets are accumulated and not played out, until the buffer utilization (BU) reaches the programmable operating point (OP) in a step **520**. Step **520** checks the condition BU $\geq$ OP, i.e. whether buffer utilization has reached or exceeded the operating point

Once the buffer utilization reaches the operating point in a step **522** ("true" in step **520**) the jitter buffer state is set to "normal", and it starts playing out the queued data, up to one byte per cycle. The packet payload, residing in multiple queues, is

multiplexing into a single TDM byte stream, by alternately playing bytes from different queues, according to the hierarchical topology of the various channels. If, at any time the buffer runs out of data, due to discontinued transmission or network disruptions, and the buffer utilization drops to 0 (step 524), the buffer state is set back to "fill" (step 526) and the data accumulation starts over.

While the invention has been described with respect to a limited number of embodiments, it will be appreciated that many variations, modifications and other applications of the invention may be made.

WHAT IS CLAIMED IS

1. A method for partitioning allocation and management of jitter buffer memory for TDM circuit emulation applications comprising the steps of:

a. obtaining a channel hierarchy for a plurality of packet carrying channels having different channel rates;

b. obtaining a packet sequential number; and

c. using said channel hierarchy and said packet sequential number, generating a base address in the jitter buffer memory;

whereby each said channel is allocated a space in the buffer memory that is proportional to the rate of said channel, and whereby out-of-order packets are automatically reordered by the jitter buffer.

2. The method of claim 1, wherein at least two of said plurality of channels include packets of different size.

3. The method of claim 1, wherein said packets are received by the jitter buffer out-of-order.

4. The method of claim 1, wherein said step of obtaining a channel hierarchy includes obtaining a channel identification number.

5. The method of claim 4, wherein said channel identification number includes 9 bits.

6. The method of claim 1, wherein said step of obtaining a packet sequential number includes obtaining a 14 bit sequential number from a packet CEP header.

7. The method of claim 1, wherein said step of generating a base address in the jitter buffer memory further includes determining an address size based on the memory size, and a byte offset size based on the size of said packet.

8. A method for partitioning a jitter buffer memory receiving a plurality of packet streams, each stream carried by a channel arranged in a given channel hierarchy, the method comprising the steps of:

a. dividing the buffer memory into a plurality of hierarchically arranged queues; and

b. allocating each said queue to one channel, so that said queue hierarchy follows the channel hierarchy.

9. The method of claim 8, wherein said dividing the buffer memory into a plurality of hierarchically arranged queues includes using power of 2 division factors.

10. A hierarchically partitioned jitter buffer memory comprising:

a. a plurality of hierarchically arranged queues correlated with a channel hierarchy; and

b. means for addressing said queues.

11. The jitter buffer memory of claim 10, wherein said queues are further divided into segments, each said segment designed to hold one packet.

12. The jitter buffer memory of claim 11, wherein each of said segments is characterized by a size in bytes correlated with a maximum packet size carried by a respective said channel.

13. The jitter buffer memory of claim 11, wherein the number of said segments in bytes is an integer power of 2.

14. The jitter buffer memory of claim 13, wherein said segment size is the minimum integer power of 2 that can hold said maximum packet size.

ABSTRACT OF THE DISCLOSURE

A method for partitioning allocation and management of jitter buffer memory for TDM circuit emulation applications comprises the steps of obtaining a channel hierarchy for a plurality of packet carrying channels having different channel rates, obtaining a packet sequential number, and using the channel hierarchy and the packet sequential number, generating a base address in the jitter buffer memory. Each channel is allocated a space in the buffer memory that is proportional to its rate, and out-of-order packets are automatically reordered by the jitter buffer.

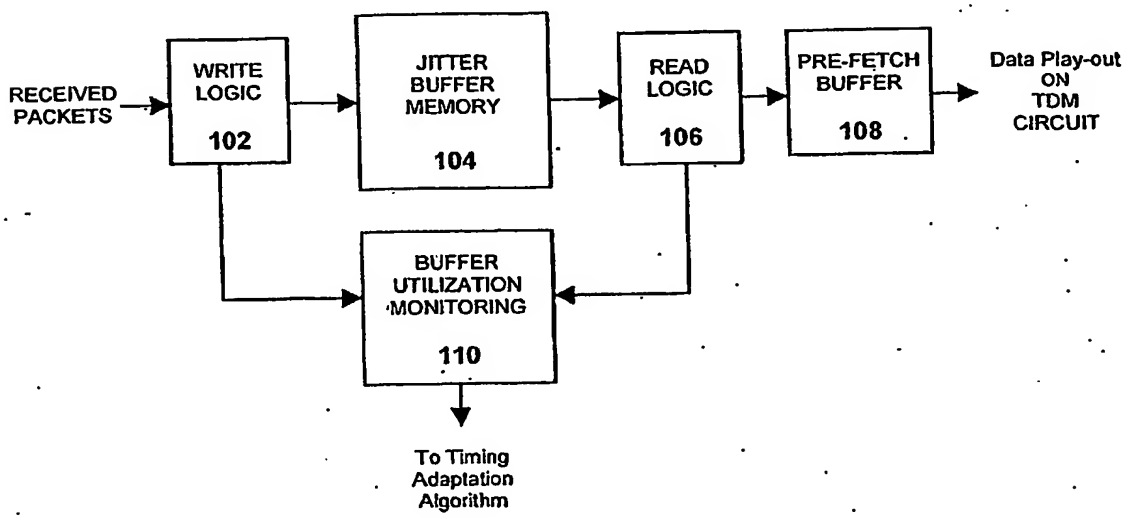
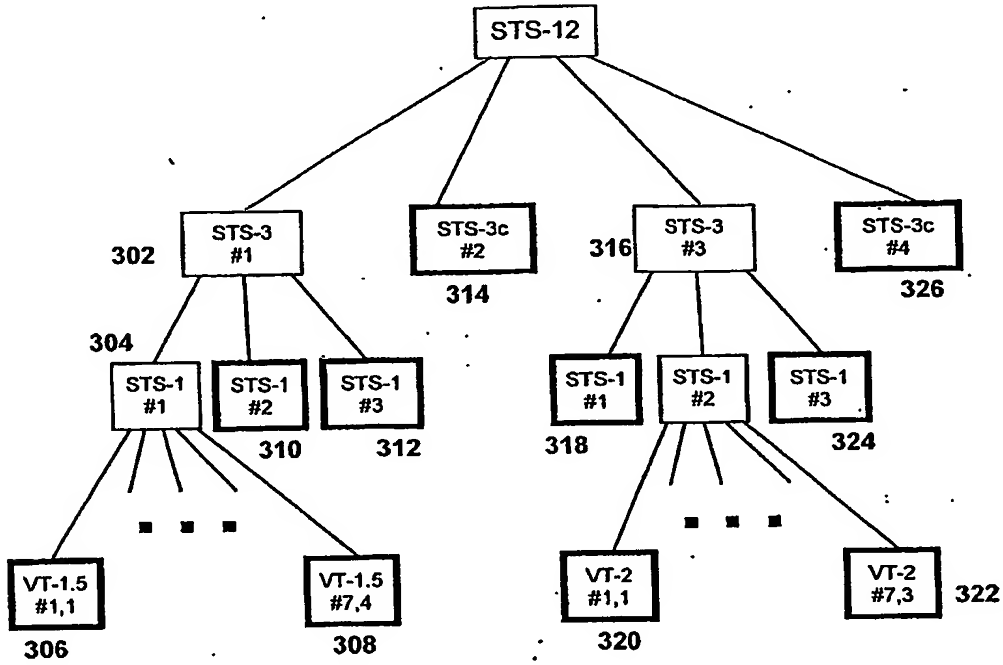

RECEIVED
PACKETS
WRITE
LOGIC
102
JITTER
BUFFER
MEMORY
104
READ
LOGIC
106
PRE-FETCH
BUFFER
108
Data Play-out
ON
TDM
CIRCUIT
BUFFER
UTILIZATION
MONITORING
110
To Timing
Adaptation
Algorithm

Figure 1



STS-12
STS-3
#1
302
STS-3c
#2
314
STS-3
#3
316
STS-3c
#4
326
304
STS-1
#1
STS-1
#2
310
STS-1
#3
312
STS-1
#1
318
STS-1
#2
STS-1
#3
324
VT-1.5
#1,1
306
VT-1.5
#7,4
308
VT-2
#1,1
320
VT-2
#7,3
322

Figure 2

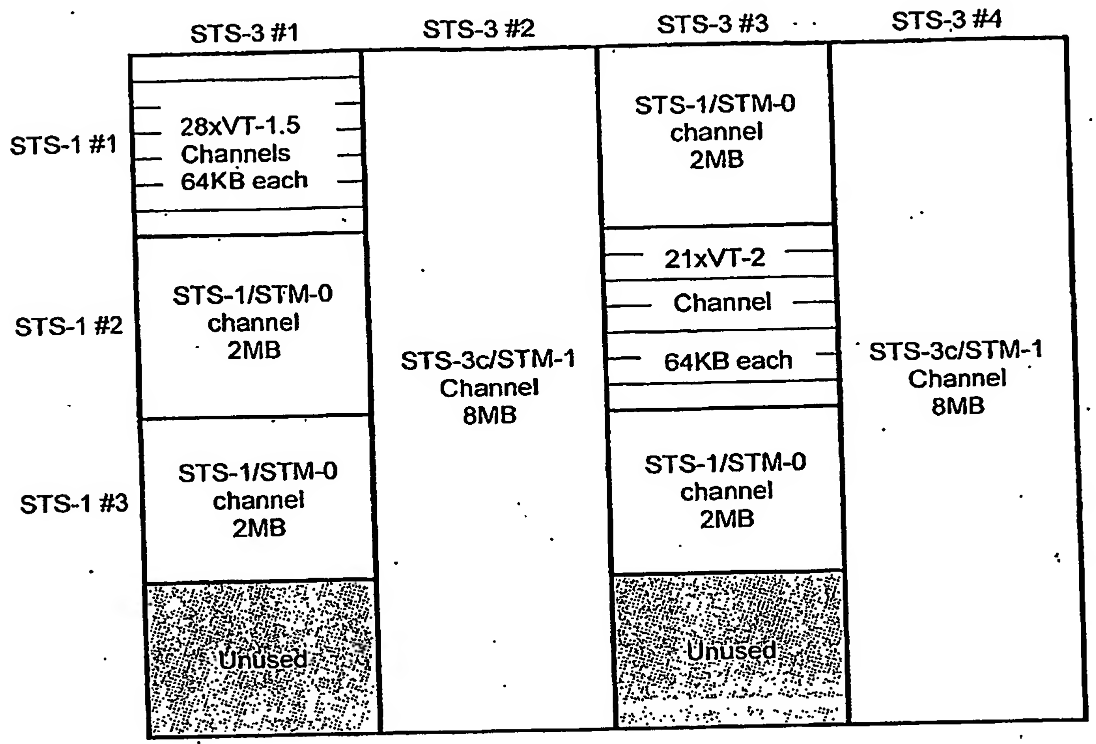

STS-3 #1
STS-3 #2
STS-3 #3
STS-3 #4
STS-1 #1
STS-1 #2
STS-1 #3
28xVT-1.5
Channels
64KB each
STS-1/STM-0
channel
2MB
STS-1/STM-0
channel
2MB
Unused
STS-3c/STM-1
Channel
8MB
STS-1/STM-0
channel
2MB
21xVT-2
Channel
64KB each
STS-1/STM-0
channel
2MB
Unused
STS-3c/STM-1
Channel
8MB

Figure 3

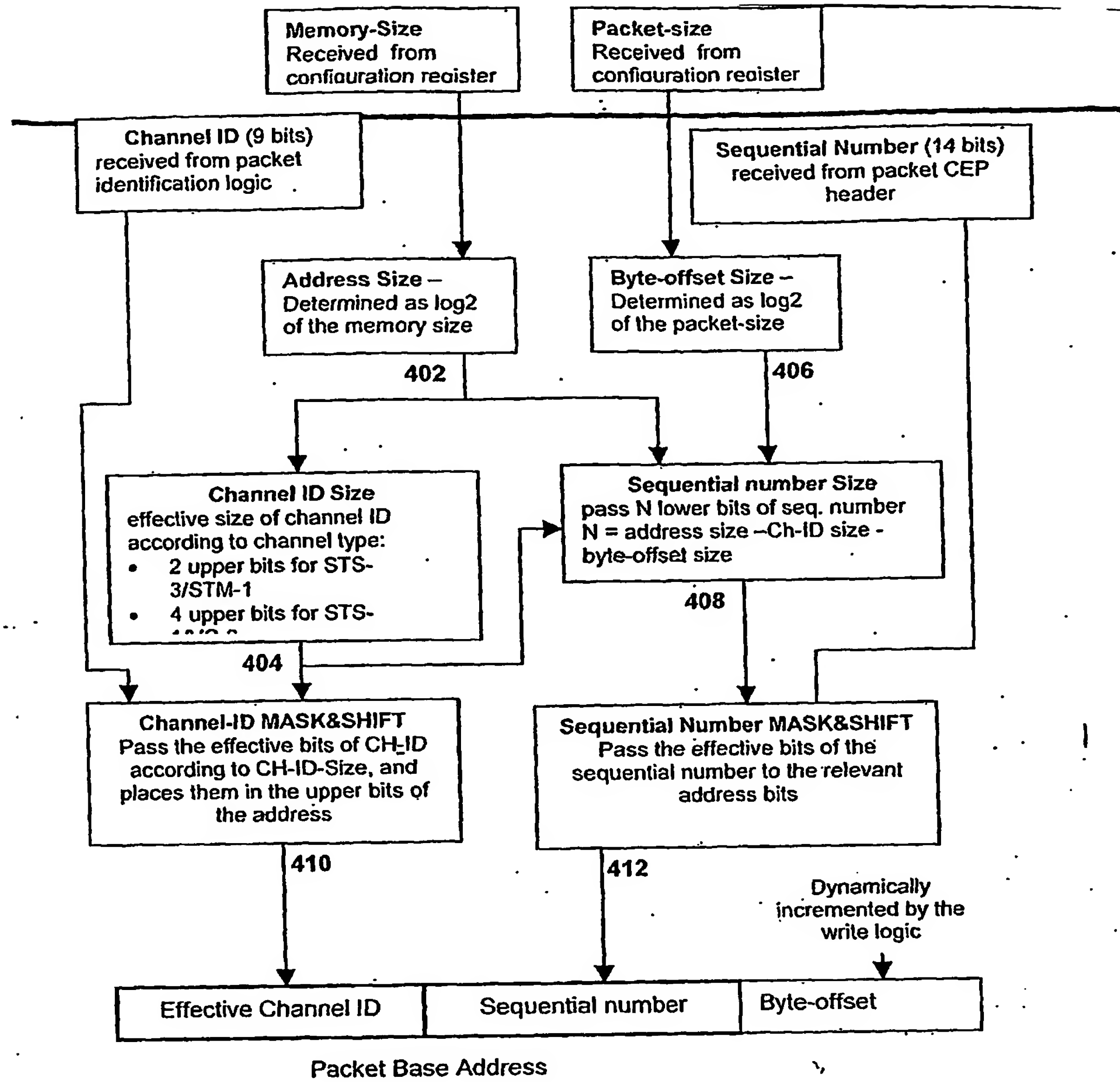

Memory-Size Received from configuration register
Packet-size Received from configuration register
Channel ID (9 bits) received from packet identification logic
Sequential Number (14 bits) received from packet CEP header
Address Size – Determined as log2 of the memory size
Byte-offset Size – Determined as log2 of the packet-size
402
406
Channel ID Size effective size of channel ID according to channel type:
• 2 upper bits for STS-3/STM-1
• 4 upper bits for STS-
Sequential number Size pass N lower bits of seq. number N = address size –Ch-ID size - byte-offset size
404
408
Channel-ID MASK&SHIFT Pass the effective bits of CH-ID according to CH-ID-Size, and places them in the upper bits of the address
Sequential Number MASK&SHIFT Pass the effective bits of the sequential number to the relevant address bits
410
412
Dynamically incremented by the write logic
Effective Channel ID
Sequential number
Byte-offset
Packet Base Address

Figure 4

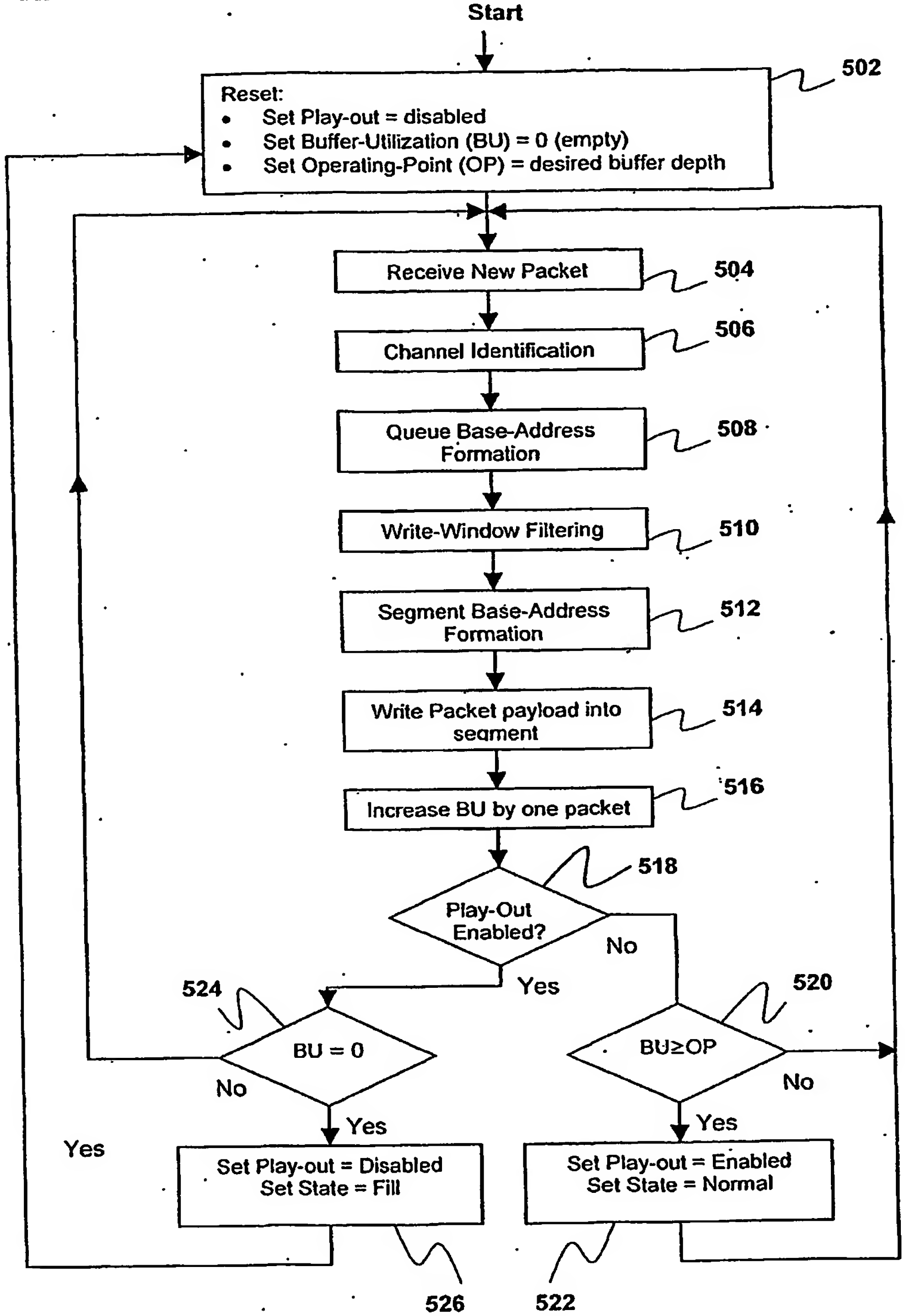

Start
Reset:
Set Play-out = disabled
Set Buffer-Utilization (BU) = 0 (empty)
Set Operating-Point (OP) = desired buffer depth
502
Receive New Packet
504
Channel Identification
506
Queue Base-Address Formation
508
Write-Window Filtering
510
Segment Base-Address Formation
512
Write Packet payload into segment
514
Increase BU by one packet
516
Play-Out Enabled?
518
No
Yes
524
BU = 0
No
Yes
Yes
520
BU≥OP
No
Yes
Set Play-out = Disabled
Set State = Fill
Set Play-out = Enabled
Set State = Normal
526
522

Figure 5